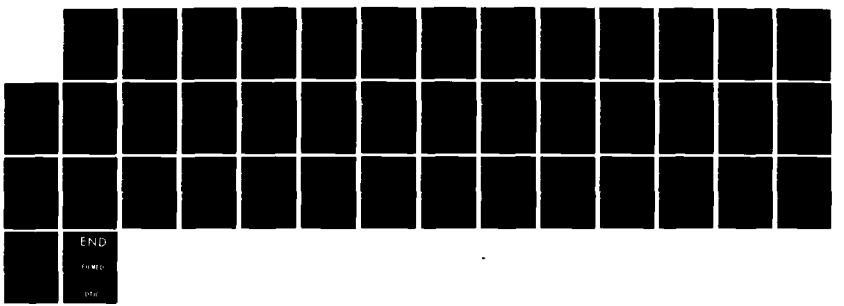
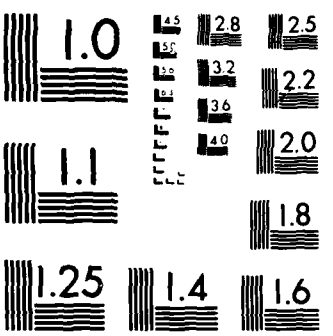
AD-A153 987 PERFORMANCE EVALUATION OF STOCHASTIC TIMED DECISION-FREE PETRI NETS(U) MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION AND D.. R P WILEY MAR 85 1/1

UNCLASSIFIED LIDS-P-1443 N00014-77-C-0532 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

March 1985 LIDS-P-1443

PERFORMANCE EVALUATION OF STOCHASTIC TIMED DECISION-FREE PETRI NETS

by

R. Paul Wiley
Robert R. Tenney

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

24th CDC and Transactions

This research was supported by the Office of Naval Research under grants ONR/N00014-77-C-0532 (NR 041-519) and ONR/N00014-84-K-0519 (NR 649-003)

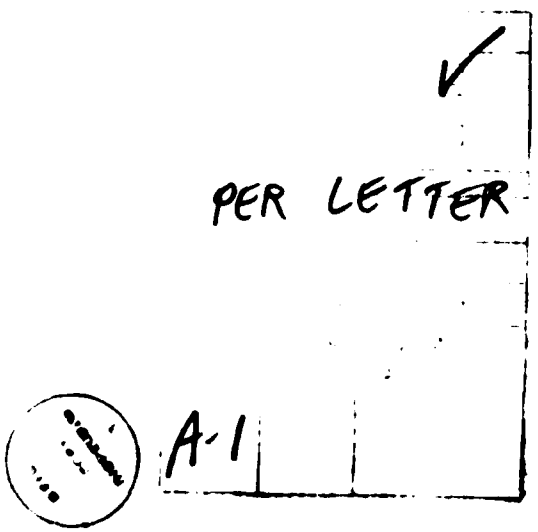
I. INTRODUCTION

The advent of low-cost processors has made the construction of highly complex, decentralized systems feasible. These systems are usually divided into interacting components which operate concurrency and coordinate their operations by using an asynchronous protocol. Petri Nets have frequently been used to model such systems since they can model concurrency and asynchronous protocols [10]. Specific examples of systems modeled by Petri Nets include computer systems [11], communication protocols [4] and manufacturing systems [5].

One aspect of designing these complex, concurrent, asynchronous systems is predicting their performance with respect to time related measures. These measures could include, but are not limited to, throughput, backlog, the probability that a shared resource is free at any given time, the average time it takes to complete each particular task, etc. In order to use Petri Nets to predict these performance measures, the notion of time must be added to the original definition. Several authors, Ramchandani [12], Sifakis [13], and Ramamoothy and Ho [11], have studied Deterministic Timed Petri Nets, where deterministic processing times have been added to model fixed delays. Stochastic Timed Petri Nets (STPNs) have been studied by Zuberek [14], Molloy [8], and Marsan, et. al. [7]. All of these authors perform the analysis by transforming the problem into an equivalent Markov Chain, whose steady state probability distribution is then found by conventional techniques. Since the numbers of states in the chain tends to grow exponentially with the size of the net, their techniques are limited to solving relatively small problems.

In this paper, we will show how the performance of a simple subclass of STPNs, that of Stochastic Timed Decision-Free Petri Nets (STDFPNs), can be predicted. This subclass can model concurrency and coordination, but not decisions, and have been used to model certain manufacturing systems [5]. The major contribution is a new analysis technique which uses a set of equations that describe the behavior of the system. (These are the same equations that Dubois and Stecke [5] used to analyze the deterministic case). The resulting algorithms are computationally more attractive and the general approach can be extended to study less restrictive classes of STPNs.

An overview of this paper goes as follows. We start by defining STPNs and presenting the relevant concepts. Then, the study is limited to STDFPNs. We examine three related processes which describe firing times, relative firing times, and intertransition times. The first process introduces the new analytical technique, the second one provides a convergence proof, and the third one allows performance measures of interest to be computed.



II. STOCHASTIC TIMED PETRI NETS

Stochastic Timed Petri Nets are graphs with two types of nodes: places (drawn as a circle and labelled p_i) and transitions (drawn as a line segment and labelled t_i), and directed arcs going from one type of node to the other (see Figure 1). Besides nodes and arcs, a STPN assigns, to every place, a nonnegative number of tokens, called the marking of the net, and a nonnegative random processing time[1]. When a token arrives at a place, it is defined to be unavailable (following Sifakis [10]). The token remains in this state until the instant when the processing has finished. At this time, the token becomes available. Initially, all tokens are assumed to be unavailable.

Tokens move around by transition firings. A transition is enabled, that is, it may fire, only when all of its input places have available tokens. When a transition does fire, it removes a token from all of its input places and adds a token to all of its output places. Two or more transitions are said to be in conflict if firing one will disable the others. If an enabled transition is not in conflict, it fires instantly. If several enabled transitions are in conflict, then a decision rule, specified a priori, selects one and that transition fires instantly. Thus, once the graph, the initial marking, processing times and decision rules are specified, the STPN evolves autonomously in time.

One can think of the operation of a STPN as a succession of markings, with the transitions from one marking to another specified by the transition

[1]Many authors, Ramchandani [12], Zuberek [14], and Molloy [8], associate processing times with transitions instead of places. The distinction makes no difference for the class of problems studied in this paper.

firings and the rules for moving tokens. If the number of tokens in any particular place can never exceed one, regardless of what processing time probability distributions and decision rules are assigned, the STPN is said to be <u>safe</u>. If there exists a τ_j for any transition t_j and for any $\varepsilon>0$ such that

$$\text{Prob}\{\text{transition } t_j \text{ will fire before } \tau_j\} > 1-\varepsilon$$

regardless of what marking the net has reached, the STPN is defined to be <u>live</u>. And finally, if there exists a directed path from any node (place or transition) to any other node, the STPN is <u>strongly connected</u>. Throughout this paper, we will assume that the STPNs we are studying are strongly connected, live, safe, and with a finite number of nodes.

III. STOCHASTIC TIMED DECISION-FREE PETRI NETS

Stochastic Timed Decision-Free Petri Nets are STPNs in which every place has exactly one input and one output transition (see Figure 2 for an example). Thus, transitions are never in conflict and decisions rules are not needed. This fact makes STDFPNs much easier to analyze than broader subclasses of STPNs, but also imposes a modeling limitation since, as noted previously, they connect model decisions. They can, however, model concurrency and coordination. The performance measures that we are interested in obtaining are the steady state firing rate of every transition, the steady state number of tokens in every place, and the steady state delay a token encounters in every place.

But before we proceed, let us review the pertinent results of Decision-Free Petri Net (DFPN) theory, or STDFPNs without the processing times. These Petri Nets, also called Marked Graphs in the literature, have been extensively studied [3], [9], with most of the work centering around establishing liveness and safeness conditions. The results that are important for our study are:

(i) The number of tokens in a directed circuit does not change as a result of transition firings.

(ii) A DFPN is live iff the number of tokens in every directed circuit is positive.

(iii) A live marking is safe iff every place is contained in a

directed circuit with a token count of one.

We will now show that these results also hold true for STDFPNs by showing that they are live and safe iff the underlying DFPNs are live and safe. The safeness equivalence is true by definition. The liveness equivalence, however, must be proved. To that effect let $z_i(k)$ be the processing time for place p_i for the k^{th} token to arrive at that place. We will assume that there exists a τ such that

$$\text{Prob}\{z_i(k) < \tau\} > 1-\varepsilon$$

for any $\varepsilon>0$, <u>regardless</u> of the numerical values of the other service times (a mild independence condition).

<u>Proposition 1</u>. Given the processing time assumption mentioned above, a STDFPN is live iff the underlying DFPN is live.

<u>Proof</u>: If the STDFPN is live, then there exists at least one sequence of transition firings and a τ such that, for any transition t_j and any $\varepsilon>0$,

$$\text{Prob}\{\text{transition } t_j \text{ will fire before } \tau\} > \varepsilon.$$

The same sequence of transition firings is valid in the underlying DFPN, which shows it is live.

Assume the underlying DFPN is live. The liveness proof given by Commoner, et.al [3] ensures that, for any particular transition t_j, there exists a circuit-free backtracking subgraph such that at least one of the

transitions, call it t_i, contains tokens in all of its input places. By the assumption on the processing times, there exists a τ_i such that, for any $\varepsilon>0$,

$$\text{Prob}\{\text{transition } t_i \text{ will fire before } \tau_i\} > 1-\varepsilon.$$

A similar argument can then be given for all other transitions along the subgraph. Thus, there exists a τ_j for any $\varepsilon>0$ such that

$$\text{Prob}\{\text{transition } t_j \text{ will fire before } \tau_j\} > 1-\varepsilon$$

which shows the STDFPN is live.

Q.E.D.

As can be noted from the listed results, directed circuits play an important role in the study of STDFPNs. It is useful to think of the operation of a STDFPN in terms of an equivalent circuit-free unfolded STDFPN. This unfolded net consists of an infinite number of identical stages, each of which is formed by breaking the regular STDFPN immediately before every place containing a token in the initial marking and "unfolding" the net. For example, Figure 3 shows the Unfolded STDFPN that corresponds to the STDFPN shown in Figure 2.

A formal procedure to construct one stage of the Unfolded STDFPN is given by Algorithm 1. In this algorithm, the variable k represents the stage and the transition firings are symbolic, not actual transition firings of the STDFPN. We also denote the set of input (output) places to

transition t_i by ${}^{\bullet}t_i$ ($t_i^{\bullet}$), as commonly done in the literature [12], [13].

Algorithm 1: Construction of one stage of the Unfolded STDFPN

INITIALIZED: M to M_0 (initial marking)

- $T_M^k \leftarrow \{t_i \mid t_i$ is enabled by M$\}$
- $P_k \leftarrow \{p_j \mid p_j$ contains a token in M$\}$
- Create a place p_j^k for all $p_j \in P_k$

DO UNTIL $T_M^k = \phi$

- For every transition $t_i \in T_M^k$
  - (a) Create a transition t_i^k
  - (b) Direct an arc from p_j^k to t_i^k for every $p_j \in {}^{\bullet}t_i$
  - (c) Create a place for every $p_j \in t_i^{\bullet}$

    If $p_j \in P_k$

    THEN

    Label the place p^{k+1}

    ELSE

    Label the place p_j^k

    $P_k \leftarrow P_k \cup p_j$

    ENDIF
- Fire all transitions $t \in T_M^k$
- Update M, $T^k{}_M$

ENDUNTIL

It can be shown by using an argument similar to the liveness proof given by Commoner, et.al. [3] that Algorithm 1 fires every transition exactly once. We also note that the marking at the end of the firing sequence is the same as the initial marking since every transition has been fired the same number of times (Murata [9], Property 2). Thus, all stages are identical except for the variable k.

Besides constructing a stage of the Unfolded STDFPN, Algorithm 1 also provides a partial order among transition firings. Before a transition can fire, it must have tokens in all of its input places. When a transition comes up in the partial order established by the algorithm, all of its input places do, in fact, contain tokens.

Visualizing the operation of STDFPN in terms of its unfolded equivalent is not only convenient, but it is also exact. We will show that the two systems are dynamically equivalent after we derive the firing time state equations, which we do next.

IV. FIRING TIME STATE EQUATIONS[2]

Given a STDFPN, a natural question to ask is the time at which the k^{th} firing of each transition will occur. To this objective, consider the STDFPN shown in Figure 4 and assume that the initial marking does not place a token in either p_1 or p_2. Then, if we let $x_i(k)$ be the time of the k^{th} firing of transition t_i,

$$x_3(k) = \max\{z_1(k) + x_1(k),\ z_2(k) + x_2(k)\} \tag{1}$$

where $z_i(k)$ is the processing time of place p_i for the k^{th} token that arrives to that place. Note that this type of equation can be written for a STDFPN even if it is not live or strongly connected.

Now, assume that the net shown in Figure 4 is part of a larger STDFPN that is strongly connected, live and safe. Equation (1) holds as long as the initial marking does not place a token in p_1 or p_2. If the initial marking places a token in p_1, but not p_2, then

$$x_3(k) = \max\{z_1(k) + x_1(k-1),\ z_2(k) + x_2(k)\} \tag{2}$$

where $x_1(0) = 0$. This equation is consistent since the liveness and safeness assumptions ensure that p_1 is contained in a directed circuit containing one token, which implies that $x_1(k-1) \leq x_3(k) \leq x_1(k)$ for all k. Similarly, if the initial marking places tokens in both p_1 and p_2, then we

[2]These equations are used by Dubois and Stecke [5], but not derived. Since they are important for this paper, we present a more formal derivation and show that they are consistent.

conclude that

$$x_3(k) = \max\{z_1(k) + x_1(k-1),\ z_2(k) + x_2(k-1)\} \tag{3}$$

In this discussion, we have considered a transition with only two input places. If there had been more, then the maximization operator of the state equation would have included more terms, but could still be easily written. Thus, firing time state equations could be obtained for every transition in a STDFPN in the manner discussed above. Figure 5 gives an example of a STDFPN and its corresponding firing time state equations.

Given the processing times, the state equations could be used to calculate the transition firing times recursively. These calculations would have to be performed in the proper order to satisfy all the causality constraints. As noted previously, any order consistent with the partial order provided by Algorithm 1 satisfies these requirements. The calculations would also have to be initialized properly. This can be done by setting $x_i(0) = 0$ for all places p_i. This is consistent with our assumption that, initially, all tokens are unavailable.

We mentioned previously that the visualization of the operation of a STDFPN in terms of its equivalent Unfolded STDFPN was not only convenient, but also exact. We now elaborate. By using the state equations, we will show that the two systems are dynamically equivalent. That is, we will show that there exists a one to one correspondence between the transition firings and processing times of the two systems such that the related transition firings occur at the same time provided the related processing times are equal. To this effect, let G be a STDFPN and UG be its corresponding

Unfolded STDFPN. Then, we specify that the initial marking in UG is a token in every place p_i^1, where p_i contains a token in the initial marking of G. Now for the proposition.

Proposition 2: The dynamic evolution of a STDFPN is equivalent to its corresponding Unfolded STDFPN.

Proof: If we associate $x_j(k)$ and $z_j(k)$ with the transition t_j^k and place p_j^k of the Unfolded STDFPN, respectively, then the state equations for the STDFPN and its corresponding Unfolded STDFPN are identical.

Q.E.D.

The dynamic behavior of a STDFPN is difficult to understand based on the state equations as given. To improve this situation, we will show that the state equations can be expanded so that the k^{th} firing time of any transition depends only on the k-1 firing times of other transitions. Then we will show that the resulting equations can be written in matrix form by converting them to a (max,+) algebra.

As seen from the derivation of the firing time state equations, the k^{th} firing of any particular transition t_j depends, in general, on the k-1 or k^{th} firings of other transitions. By repeatedly substituting the k^{th} firings times of these other transitions, the equations can be expanded so that they depend only on the k-1 firing times of other transitions and on the k^{th} service times of several places. The repeated substitution procedure corresponds to tracing backwards along all paths in the Unfolded STDFPN, starting from t_j^k and ending at transitions in the previous stage.

Since the Unfolded STDFPN contains no directed circuits, these backtracking paths are guaranteed to end in the previous stage and are finite. And all places that appear along the path correspond to the k^{th} stage of the Unfolded STDFPN. Therefore only $z_i(k)$ terms appear in the state equations for $x_j(k)$, and no $z_i(m)$, $m<k$, terms appear.

To make the structure of these equations clearer, substitute the max operation by $\oplus$ and + by $\odot$, as done in [5]. If we let $\underline{x}(k)$ be the vector for the k^{th} firing time of all transitions, then the discussion on repeated substitution and some algebraic manipulations shows that

$$\underline{x}(k) = A[\underline{z}(k)] \odot \underline{x}(k-1) \qquad (4)$$

where A[z(k)] is a matrix whose components are functions of $\underline{z}(k)$, the vector of processing time variables, and the matrix multiplication is in the sense of $\oplus$ and $\odot$ operators. The assumption that all tokens are initially unavailable implies that $\underline{x}(0) = \underline{0}$, the vector of all zeroes.

Cohen, et.al. [2] analyzes systems which have state equations of the form given in (4), but with deterministic processing times. Dubois and Stecke [5] take this approach to analyze Deterministic Timed DFPNs. We will also use some of their results.

So far, in this section, we have spent a lot of time obtaining and clarifying the structure of the firing time state equations. Since the processing times are, in general, random variables, it is necessary that we try to characterize the probability distribution for $\underline{x}(k)$, which we denote by $f_{\underline{x}(k)}(\underline{x})$. To keep the calculations from getting arbitrarily complex, we assume, from this point on, that the processing times are independent random

variables over the stages. With this assumption, we can show the following.

Proposition 3. If the $\underline{z}(k)$ are independent random vectors over k, and indepenent from $\underline{x}(0)$, then the process $\{\underline{x}(k), k \in N\}$ is Markov.

Proof: Follows from equation (4) and the independence assumptions.

Q.E.D.

Thus, the vector of firing times, $\underline{x}(k)$, constitutes the state of a Markov Process. This conclusion is intuitive if one realizes that the transitions whose firing times are included in $\underline{x}(k)$ constitute a cut set in the Unfolded STDFPN. The Markov property also implies that one can use the firing time state equation to calculate $f_{\underline{x}(k)}(\underline{x})$ starting from $f_{\underline{x}(0)}(x)$ for all k. This, however, may not be the indicated course of action. The variance for any particular $x_i(k)$ will increase without bound as k increases, as we can convince ourselves by considering a very simple STDFPN containing one transition and one place connected in a loop and with a random processing time. Furthermore, we know of no way in which to characterize $f_{\underline{x}(k)}(\underline{x})$ as k goes to infinity by using some central limit theorem or assuming some clever probability for the processing times.

Thus, we cannot use $f_{\underline{x}(k)}(\underline{x})$ to obtain the desired performance measures since the probability distributions do not converge. To correct this situation, we will do a few things. First, we will derive the relative firing times process and show that the probability distributions that characterize it will converge by making some additional assumptions on the

processing times. Then, we will define the intertransition times process, show that the probability distributions that characterize it will converge by relating it to the relative firing times. And finally, we will show how the performance measures can be obtained from the steady state probability distributions of the intertransition times.

V. RELATIVE FIRING TIMES

The relative firing times are obtained by performing a transformation on the firing times. To that effect, pick a transition t_r to be the reference transition. Then $\underline{w}(k)$, the vector of relative firing times, is defined by

$$\underline{w}(k) = \underline{x}(k) - x_r(k)\underline{1} \tag{5}$$

where $\underline{1}$ is a vector of ones and $x_r(k)$ is the k^{th} firing time of the reference transition. We can also represent this transformation in (max, +) algebra as follows

$$\underline{w}(k) = [x_r(k)]^{-1} \odot \underline{x}(k) \tag{6}$$

where $[\cdot]^{-1}$ denotes subtraction in the usual algebra. As can be seen from this last equation, the transformation performed on $\underline{x}(k)$ to obtain $\underline{w}(k)$ is a scaling operation in the (max,+) algebra domain. In the paragraphs that follow, to clarify the discussion, we let

$$\underline{w}(k) = S[\underline{x}(k)] \tag{7}$$

represent the transformation indicated by equations (5) and (6).

In order for this transformation from $\underline{x}(k)$ to $\underline{w}(k)$ to be useful, it must be well defined. That is, we must show that the derivation of $\underline{w}(k+1)$ from $\underline{x}(k+1)$ using the definition is equivalent to deriving $\underline{w}(k+1)$ from $\underline{w}(k)$ using a state equation for the relative firing times process. We do this

next.

Proposition 4. The transformation from $\underline{x}(k)$ to $\underline{w}(k)$ is well defined, that is,

$$S[x(k+1)] = S\{A[z(k)] \odot S[x(k)]\} \tag{8}$$

Proof[3]: By the definition of relative firing times, we have that

$$S[x(k+1)] = A[z(k)] \odot \underline{x}(k) \odot \{\underline{a}_r^T[\underline{z}(k)] \odot \underline{x}(k)\}^{-1} \cdot x_r(k) \odot x_r^{-1}(k)$$

$$= A[z(k)] \odot \underline{x}(k) \odot x_r^{-1}(k) \odot \{\underline{a}_r^T[z(k)] \odot \underline{x}(k) \odot \underline{x}_r^{-1}(k)\}^{-1}$$

$$= A[z(k)] \odot S\{\underline{x}(k)] \odot \{\underline{a}_r^T\underline{z}(k)] \odot S[\underline{x}(k)]\}^{-1}$$

$$= S\{A[\underline{z}(k)] \odot S[\underline{x}(k)]\} \tag{10}$$

which is what we wanted to show.

Q.E.D.

Expanding the last equation, we obtain the state equation for the relative firing times process

[3]This proof uses (max,+) algebra results which can be verified by converting back to the usual algebra or by consulting [2].

$$\underline{w}(k+1) = A[\underline{z}(k)] \odot \underline{w}(k) \odot \{\underline{a}_r^T[\underline{z}(k)] \odot \underline{w}(k)\} \tag{11}$$

With this state equation, we can prove the following proposition.

Proposition 5. If the $\underline{z}(k)$ are independent random vectors over k, and independent from $\underline{w}(0)$, then the process $\{\underline{w}(k), k \in N\}$ is Markov.

This conclusion is not surprising if we remember that the transitions whose relative firing times are contained in $\underline{w}(k)$ form a cut set of the Unfolded STDFPN. The transformation $S(\cdot)$ subtracts the absolute value of time, and this has no effect on the Markov property. The transformation does, however, provide us with some nice convergence properties, as we will see next.

VI. PROPERTIES OF THE RELATIVE FIRING TIMES MARKOV PROCESS

In this section, we analyze the consequences of making further assumptions on the processing times. First, if we assume that the processing times are bounded, we will show that the state space is bounded. In turn, if we assume the processing time probability distributions are stationary and in discrete time, we will show that the resulting Markov process has a single recurrence class. Then, we will combine all of the assumptions stated above, and analyze the consequences.

So, first assume that the service times are bounded for all places and for all k. By incorporating this assumption into the liveness proof given by Commoner, et.al. [3], we can show that the time to fire any particular transition, from all possible markings, is bounded. Call this bound τ. We can also show that a strongly connected STDFPN can be considered as the union of a finite number of directed circuits, each containing one token. This, in turn, leads us to conclude that the number of times any particular transition fires differs from the number of times any other transition fires by less than some finite number. Let this number be L. Now, consider the STDFPN at any instant of time when the reference transition t_r fires, say for the k^{th} time. Let t_j be any other transition. We know that the number of times transition t_j has fired differs from k by at most L. We also know that the time between t_j firings is at most τ. Therefore

$$|x_r(k) - x_j(k)| \leq \tau L \tag{12}$$

Thus, every element of $\underline{w}(k)$ is bounded. We have just shown the following:

Proposition 6. If all elements of $\underline{z}(k)$ are bounded for all k, then all elements of $\underline{w}(k)$ are bounded for all k.

Now assume that the processing times $\underline{z}(k)$ are identically distributed over k and in discrete time. If a value of $\underline{Z}$ for which $f_{\underline{z}}(\underline{Z}) > 0$ occurs enough times in a row, then the results of Cohen, et.al. [2] tell us that the firing times of all the transitions will become periodic modulo a constant, regardless of what numerical value the initial firing times had. That is, for any $\underline{x}(0)$, there exists finite k_0, λ and d such that

$$\underline{x}(k+d) = d\lambda 1 + \underline{x}(k) \qquad (13)$$

for all $k \geq k_0$. The x(k)'s related by this equation are, in fact, the same state in the relative firing times Markov process. Thus, if $\underline{Z}$ occurs k_0 times in a row, an event which occurs with probability $[f_{\underline{z}}(Z)]^{k_0} > 0$, the relative firing times Markov process will be in a set of periodic states, and will remain in these states as long as the value of $\underline{Z}$ keeps occurring.

To analyze the consequences of making the specified assumptions further, let us define deterministic states to be the states of the relative firing times Markov process which can be reached by repeating any particular value of $\underline{Z}$ enough times. These deterministic states are reachable from anywhere in the state space and, in particular, from each other. Now we can prove the following.

Proposition 7. If $\underline{z}(k)$ is independent and identically distributed over k, then the discrete-state Markov Process $\{\underline{w}(k), k \in N\}$ has a single recurrence class.

Proof: Assume more than one recurrence class exists. From the previous discussion, each recurrence class must include the deterministic states. This is a contradiction since separate recurrence classes do not have states in common. Therefore, the $\{\underline{w}(k),\ k \in N\}$ Markov Process has a single recurrence class.

Q.E.D.

In this subsection, we have shown that the $\underline{w}(k)$ state space is bounded if the $\underline{z}(k)$ are bounded for all k. If we also assume that the $\underline{z}(k)$ are in discrete time, then we conclude that the number of $\underline{w}(k)$ states is finite. And finally, if we also assume that the $\underline{z}(k)$ are stationary, we conclude that the resulting $\{\underline{w}(k),\ k \in N\}$ discrete-state discrete-index Markov process has a single recurrence class and a finite number of states. Examples exist, however, of STDFPNs whose relative firing times Markov process has periodic states. Therefore, we cannot conclude that this process has an ergodic probability distribution. But we can conclude that

$$F(\underline{W}) = \lim_{k\to\infty} \frac{1}{k} \sum_{L=1}^{k} f_{\underline{w}(L)}(\underline{W})$$

$$= \lim_{k\to\infty} \frac{1}{d} \left[f_{\underline{w}(kd)}(\underline{W}) + f_{\underline{w}(kd+1)}(\underline{W}) + \ldots + f_{\underline{w}(kd+d-1)}(\underline{W}) \right] \qquad (14)$$

where d is the period of the Markov chain, converges [1]. Furthermore, $F(\underline{W})$ is equal to the fraction of time the process spends in state $\underline{W}$, that is

$$F(\underline{w}) = \lim_{k\to\infty} \frac{1}{k} \sum_{L=1}^{k} \delta(\underline{w}(L)-\underline{w}) \tag{15}$$

where $\delta(\cdot)$ is the unit sample function. That is, $F(\underline{W})$ is the unique state occupation distribution.

Thus, we have shown that the relative firing times Markov process has a unique state occupation distribution. The problem remains, however, of how to calculate the performance measures. We will obtain these measures from the intertransition times, which we define next. We will show that the intertransition times have a unique state occupation distribution by relating them to the relative firing times.

VII. INTERTRANSITION TIMES STATE EQUATIONS

An intertransition time is the time between the firings of two transitions that have one place in between them, that is, it is the time a token spends in a place. Consider the STDFPN shown in Figure 4, which we assume is part of a larger, live and safe, STDFPN. If we assume that the initial marking does not place a token in p_1, then the time between the kth firings of transitions t_1 and t_3 is given by

$$v_{13}(k) = x_3(k) - x_1(k)$$

$$= \max\{z_1(k),\ z_2(k) + x_2(k) - x_3(k-1) + x_3(k-1) - x_1(k)\} \qquad (16)$$

where we have used the firing time state equations and added a zero to one of the terms in the maximization operator. Now, place p_2 is contained in a directed circuit containing one token. By using this fact, we can conclude that $x_2(k) - x_3(k-1)$ can be expressed as a sum of intertransition times along the specified directed circuit. Similarly, $x_1(k) - x_3(k-1)$ can be expressed as a sum of intertransition times by noting that place p_1 is contained in a directed circuit with one token. Thus, we conclude that $v_{13}(k)$ can be expressed in terms of the processing times corresponding to the input places of a transition t_3, and in terms of the intertransition times along circuits with one token containing transition t_3.

The reasoning used above can be used to write intertransition time state equations among any pair of transitions. More input places can be handled by using the same argument given above several times. And if place p_1 contains a token in the initial marking, then the definition of the

intertransition changes slightly to become

$$v_{13}(k) = x_3(k) - x_1(k-1) \tag{17}$$

The rest of the reasoning is identical.

As an example, consider the STDFPN shown in Figure 5. The state equations for the four possible intertransition times are

$$v_{11}(k) = \max\{z_1(k),\ z_3(k) + v_{12}(k-1)\} \tag{18}$$

$$v_{21}(k) = \max\{z_3(k),\ z_1(k) - v_{12}(k-1)\} \tag{19}$$

$$v_{12}(k) = \max\{z_2(k),\ z_4(k) - v_{21}(k)\} \tag{20}$$

$$v_{22}(k) = \max\{z_4(k),\ z_2(k) + v_{21}(k)\} \tag{21}$$

As with the firing time state equations, the intertransition times can be calculated recursively by doing the calculations in the proper order and by initializing the process properly. The proper order can be achieved by using the partial order established by Algorithm 1. When a transition t_j is fired by that algorithm, all the information required to calculate $v_{ij}(\cdot)$ is known for all possible t_i. To initialize the intertransition time state equations, one must refer back to the original definitions and remember that $\underline{x}(0) = \underline{0}$.

Since the intertransition times can be calculated recursively, we can conclude that there exists a transformation $g(\cdot)$ such that

$$\underline{v}(k) = g\{\underline{z}(k), \underline{v}(k-1)\} \tag{22}$$

where $\underline{v}(k)$ is the vector of intertransition times which that $[\underline{v}(k)]_j = v_{ij}(k)$ if place p_j is between transitions t_i and t_j. The exact nature of $g(\cdot)$ is very complicated, but the existence of such a transformation is important since it can be used to show the following:

Proposition 8. If the $z(k)$ are independent random vectors over k, and independent from $\underline{v}(0)$, then the $\{\underline{v}(k), k \in N\}$ process is Markov.

We derived the intertransition times since the firing times process did not converge and since we could not obtain the performance measures directly from the relative firing times. Next we will show that the state occupation distributions for the intertransition times will converge by relating them to the relative firing times, and then we will show how the performance measures can be calculated.

VIII. TRANSFORMATION FROM RELATIVE TO INTERTRANSITION FIRING TIMES

From the previous section, we note that the intertransition time vector, $\underline{v}(k)$, is defined in terms of $\underline{x}(k)$ and $\underline{x}(k-1)$. This implies that for every transition from $\underline{x}(k-1)$ to $\underline{x}(k)$, $\underline{v}(k)$ is uniquely determined. We will now show that something similar occurs with the intertransition times.

Consider two connected transitions, t_1 and t_3, of Figure 4. There are two possible definitions of the intertransition times, given by equations (16) and (17). If equation (16) applies, then

$$v_{12}(k) = x_2(k) - x_r(k) - [x_1(k) - x_r(k)]$$

$$= w_2(k) - w_1(k) \tag{23}$$

If equation (17) holds, then we can show by representing the equation in (max, +) form, using firing time state equation (4), and remembering the definition of relative firing times, that

$$v_{12}(k) = w_2(k) \odot \{w_1(k-1) \odot [\underline{a}_r^T[\underline{z}(k)] \odot \underline{w}(k-1)]^{-1}\}^{-1} \tag{24}$$

Thus, the intertransition times can be calculated by knowing $\underline{w}(k-1)$, $\underline{w}(k)$, and $\underline{z}(k)$.

So, analogous to the situation we had with the firing times process, for every state transition in the $\{\underline{w}(k),\ k \in N\}$ Markov Process, the intertransition time vector $\underline{v}(k)$ is uniquely determined. This implies that the state occupation distribution for the intertransition times, $F(\underline{v})$, converges since, as we saw previously, the state occupating distribution for

the relative firing times converges.

IX. PERFORMANCE MEASURES

In the previous section, we showed that the state occupation distribution for the intertransition times process converges. Now we will show how to evaluate the performance measures of interest.

The easiest performance measure to calculate is γ_j, the average time a token spends in place p_j. It is simply the average of the intertransition time corresponding to place p_j. Thus, γ, the vector of average holding times is

$$\gamma = \sum_{\underline{v}} v \, F(v) \tag{25}$$

To find λ_j, the average firing rate of transition t_j, we must think about the way live and safe STDFPNs operate. Due to the liveness and safeness assumptions, every transition is contained in a directed circuit containing one token. Thus, the average time between firings of transition t_j, which we denote by D_j, is given by the sum of the average intertransition times around the directed circuit containing one token. The average firing rate is then given by

$$\lambda_j = \frac{1}{D_j} \tag{26}$$

To make matters simpler, the consistency result for strongly connected STDFPNs [13] implies that the average firing rate for all transitions is equal. Therefore, all we have to do to find the average firing rate is sum the average intertransition times along any directed circuit and invert.

The last performance measure of interest for STDFPNs is N_j, the average number of tokens place p_j contains, for all places. This statistic can be computed by using Little's formula [6] to obtain

$$N_j = \lambda \gamma_j \tag{26}$$

where λ is the average rate of arrival of tokens to p_j and γ_j is the average time a token sends in p_j.

X. EXAMPLE

Consider the STDFPN of Figure 5, with the processing time $z_1(k)$ being 1 or 3 with equal probability, $z_4(k) = 2$ and $z_2(k) = z_3(k) = 0$. We assume these processing times are stationary and independent from place to place and over stages. The intertransition times are given by equations (18) through (21). As can be noted from these equations, it is only necessary to iterate between $p_{v_{21}(k)}(v)$ and $p_{v_{12}(k)}(v)$. Doing this, we obtain the limiting distributions shown in Figure 6. From these distributions, we find that

$$\underline{\gamma}^T = [13/6 \quad 7/6 \quad 1 \quad 13/6], \qquad \lambda = \frac{6}{13}, \quad \text{and}$$

$$\underline{N}^T = [1 \quad \frac{7}{13} \quad \frac{6}{13} \quad 1],$$

XI. CONCLUSIONS

We have shown, in this paper, that the performance of a STDFPN can be analyzed by recursively calculating the probability distributions for the intertransition times, and repeating these calculations until the state occupation distributions converge. Of particular note is the development of the Unfolded STDFPN. This unfolded net rids the problem of the complications arising from directed circuits and provides intuition for the characterization of the different processes. Moreover, it is a concept that is likely to extend to broader subclasses of STPNs.

Finding the steady state performance measures using the intertransition times state equations is more flexible than previous approaches for solving STPNs, which consisted of transforming the problem into an equivalent Markov Chain. The mentioned state equations are valid regardless of what processing time distributions are assigned. Thus, one could, for example, solve the problem by starting with gross approximations to the given processing time distributions, perform the recursive computations until convergence, then use the results to start a new set of recursive computations using better approximations to the processing times distributions[4]. Such a procedure is not possible using previous approaches since, every time a processing time distribution is changed, a new Markov Chain has to be constructed. It is also hoped that solving these state equations recursively is numerically more stable than solving the large sets of linear equations associated with the Markov Chains.

[4]This idea is due to Ms. E.L. Hahne, whose cooperation is greatly appreciated.

There are several possible areas of future research. Extending the analysis to broader classes of problems is one. The efficient numerical implementation of the resulting algorithms, an issue not addressed in this paper, is another. On a longer term, one can think of performing a sensitivity analysis, doing aggregation studies, hierarchical decomposition, etc. STPNs seem to be an ideal area in which to study many topics of interest in system theory.

REFERENCES

[1] E. Cinlar, Introduction to Stochastic Process, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1975).

[2] G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot, "A Linear-System-Theoretic View of Discrete Event Processes", Proc. of the 22nd IEEE Conf. on Dec. and Control, Vol. 3 (December 1983), pp. 1039-1044.

[3] F. Commoner, A.W. Holt, S. Even, and A. Pnueli, "Marked Directed Graphs", Journal of Computer and System Sciences, Vol. 5 (1971), pp. 511-523.

[4] M. Diaz, "Modeling and Analysis of Communication and Cooperation Protocols using Petri Net Based Model", Computer Networks, Vol. 6, No. 6 (December 1982), pp. 419-441.

[5] D. Dubois and K.E. Stecke, "Using Petri Nets to Represent Production Processes", Proc. of the 22nd IEEE C.D.C., Vol. 3 (December 1983), pp. 1062-1067.

[6] L. Kleinrock, Queueing Systems - Vol. 1: Theory, John Wiley and Sons, New York, N.Y. (1975)

[7] M.A. Marsan, G. Conte, and G. Balbo, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems", ACM Transactions on Computer Systems, Vol. 2, No. 2 (May 1984), pp. 93-122.

[8] M.K. Molloy, "Performance Analysis Using Stochastic Petri Nets", IEEE Trans. on Computer, Vol. C-31, No. 9 (September 1982), pp. 913-917.

[9] T. Murata, "Circuit Theoretic Analysis and Synthesis of Marked Graphs", IEEE Trans. on Circuits and Systems, Vol. CAS-24, No. 7, (July 1977), pp. 400-405.

[10] J.L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, N.J. (1981)

[11] C.V. Ramamoorthy and G.S. Ho, "Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets," IEEE Trans on Software Eng., Vol. SE-6, No. 5 (September 1980), pp. 440-449.

[12] C. Ramchandani, "Analysis of Asynchronous Concurrent Systems by Timed Petri Nets", Technical Report 120, Lab. for Computer Science, M.I.T., Cambridge, MA. (1974).

[13] J. Sifakis, "Use of Petri Nets and Preliminary Performance

Evaluation", The 7th Annual Symposium on Computer Arquitecture - Conferene Proceedings, (May 6-8, 1980), pp. 88-96.

[14] W.M. Zuberek, "Timed Petri Nets and Preliminary Performance Evaluation", The 7th Annual Symposium on Computer Arquitecture - Conference Proceedings (May 6-8, 1980), pp. 88-96.

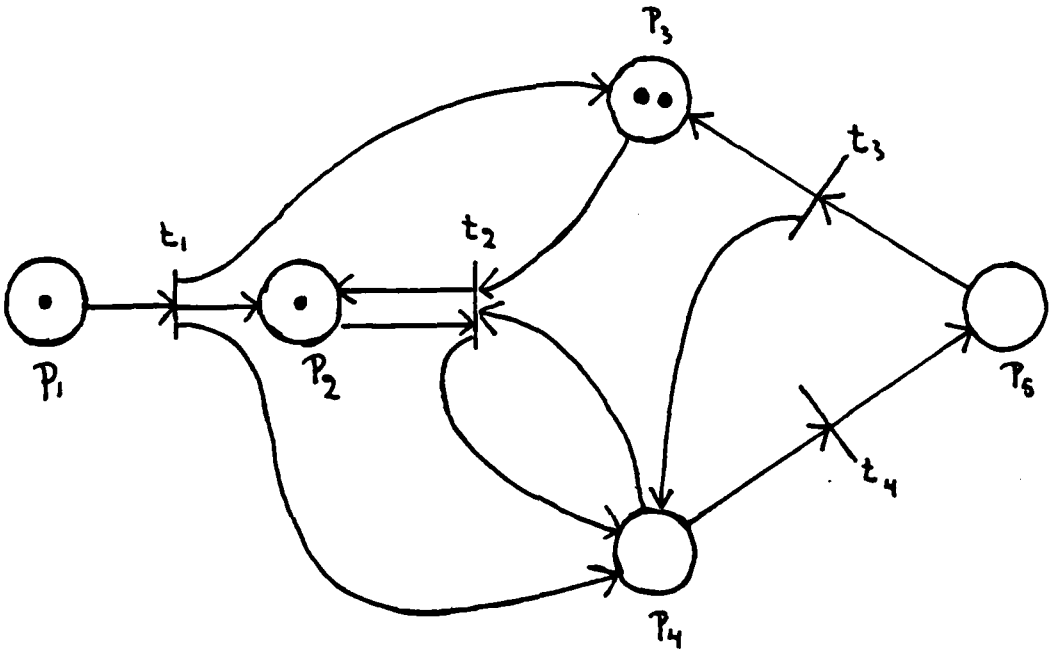


Figure 1. Stochastic Timed Petri Net Example (without processing times)

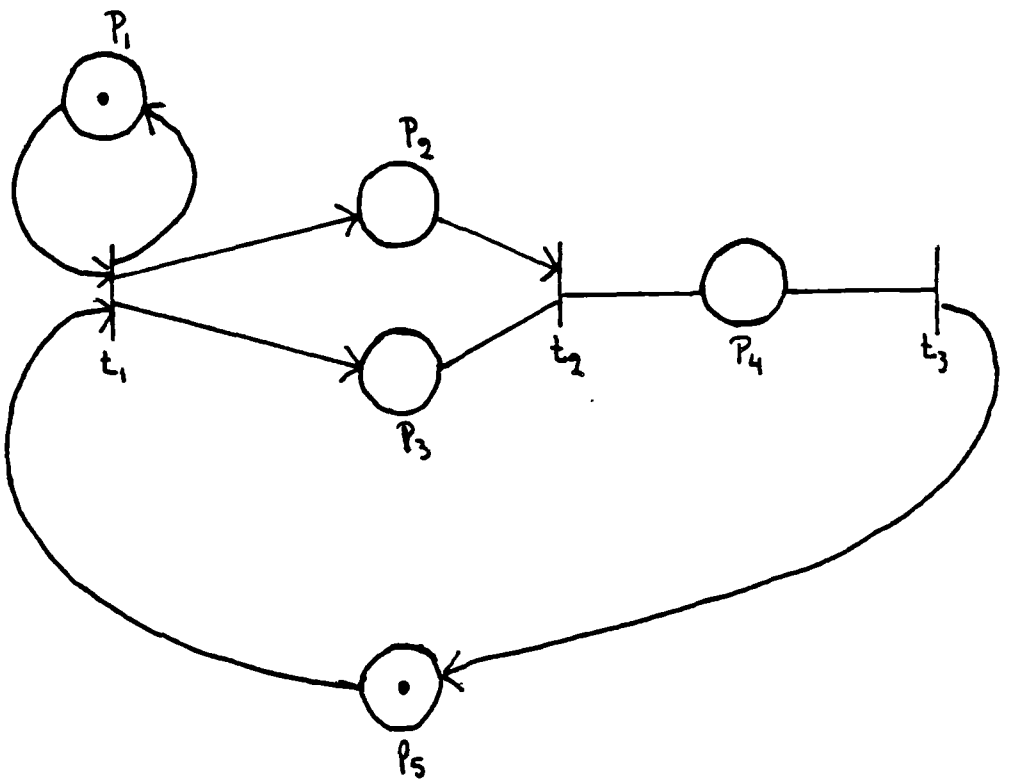


Figure 2. Stochastic Timed Decision-Free Petri Net Example (without processing times)

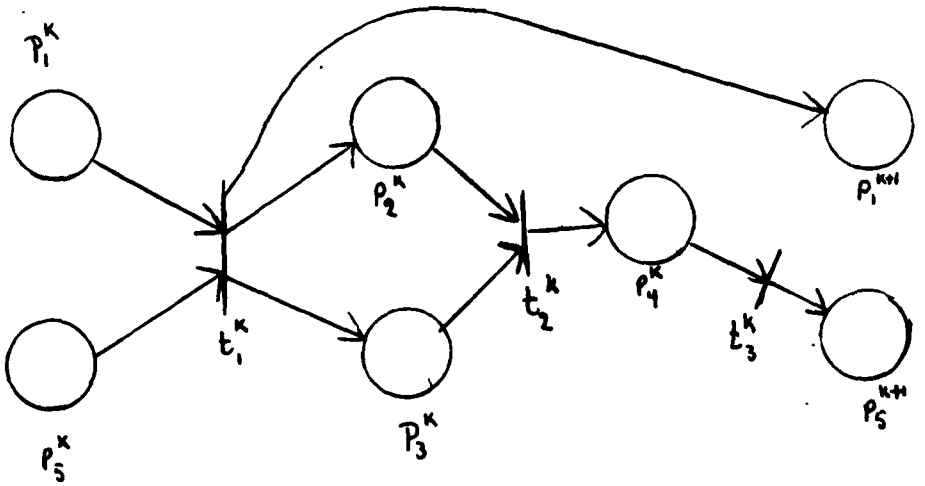


Figure 3. Unfolded STDFPN that corresponds to STDFPN of Figure 2.

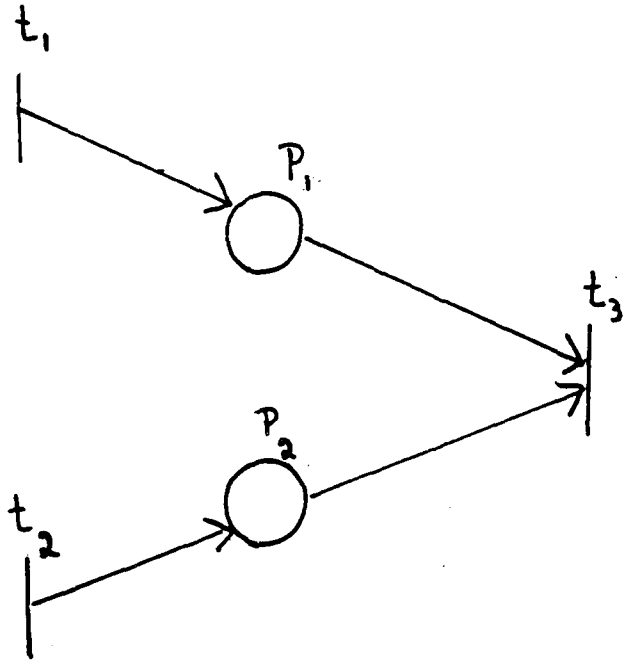
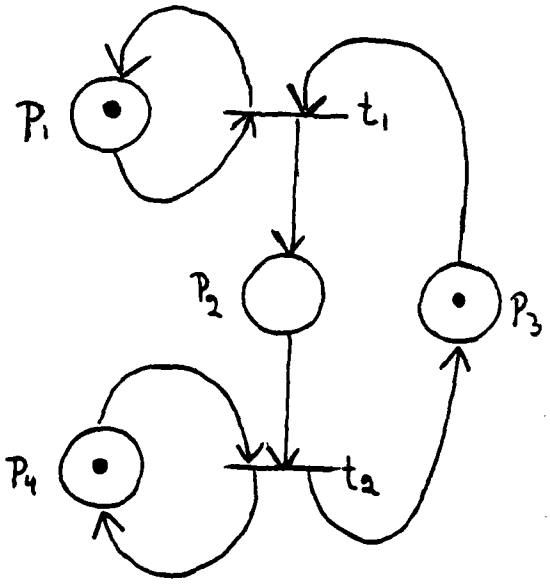


Figure 4. STDFPN for Firing Time State Equation derivation



$$x_1(k) = \max \{ z_1(k) + x_1(k-1),\ z_3(k) + x_2(k-1) \}$$

$$x_2(k) = \max \{ z_2(k) + x_1(k),\ z_4(k) + x_2(k-1) \}$$

Figure 5. STDFPN Example with Firing Time State Equations

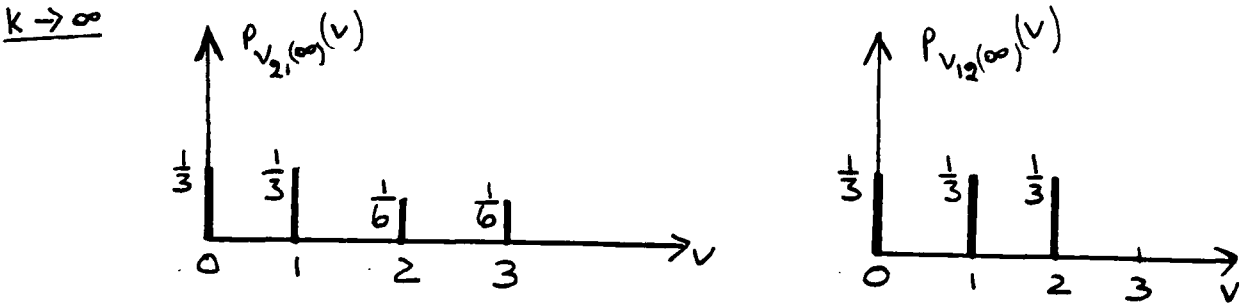


Figure 6. Limiting Probability Distributions for Numerical Example.

END

FILMED

6-85

DTIC